



11/00 20060
#18/ Supplemental 268
Arnold D w/ Appendix
R. Morgan
6/6/94

780.29767X00

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Applicants: Thomas J. CAMPANA, JR. et al
Serial No.: 07/702,938
Filed: May 20, 1991
For: SYSTEM FOR INTERCONNECTING ELECTRONIC
MAIL SYSTEMS BY RF COMMUNICATIONS
AND METHOD OF OPERATION THEREOF
Group: 2608
Examiner: G. Oehling

RECEIVED
94 JUN -3 PM 3:46
GROUP 260

SECOND SUPPLEMENTAL AMENDMENT

Honorable Commissioner of
Patents and Trademarks
Washington, D. C. 20231

May 23, 1994

Sir:

This is supplemental to the Supplemental Amendment filed
April 29, 1994.

IN THE SPECIFICATION:

Please amend the specification as follows:

✓ Page i (before Page 1), line 12, change "10-14" to
--10-12--.

In the Attached Appendix: Delete the original 15 pages
of the Appendix as filed on May 20, 1991 and insert a
substitute
Appendix which is attached to this Second Supplemental
Amendment consisting of a cover page and numbered pages 1-12.

IN THE CLAIMS:

Please add new claim 86 as follows:

~~86. A method of transferring information from a RF receiver to a processor under control of a program stored by the processor comprising:~~

~~transmitting the information with a RF transmitter to the RF receiver;~~

~~the RF receiver signalling the processor on a transmission medium of the processor used for transmission of information by the processor that the received information is stored within memory of the receiver;~~

~~controlling the transfer of the stored information from the memory of the receiver to a memory of the processor on the transmission medium with the program; and~~

~~processing the information in the memory of the processor with an application program stored in the memory of the processor.~~

REMARKS

The Applicants are presenting claim 86 herein which is identical to at least claim 81 of Serial No. 07/702,319 and claim 176 of Serial No. 07/702,939 for the purpose of having the Examiner consider the possibility of double patenting between claims 78, 79 and 81-89 in Serial No. 07/702,319 and claims 160-161 and 176 in Serial No. 07/702,939.

If the Examiner finds the subject matter of at least claim 86 to be present a situation requiring restriction, it is requested that the Examiner indicate in this application on the record to that effect and in Serial Nos. 07/702,319 and 07/702,939, respectively, with respect to claims 78, 79 and 81-89 and 160, 161 and 176. If restriction is required in this application and in Serial Nos. 07/702,319 and 07/702,939 regarding at least the subject matter of claim 86, Applicants will cancel the subject matter so restricted from all applications now pending and file a divisional application.

Newly submitted claim 86 covers the operation of the receiver 119 transferring originated information to the destination processors, as illustrated in Fig. 10, as described on pages 43 and 44 of the specification and pages 1-9 of the Appendix. Claims 68 and 69, which are respectively dependent on claims 137 and 157, also recite aspects of the transfer of the other originated information for use with application programs.

The Examiner, for purposes of determining the patentability of claim 86, is referred to the description of the prior art including the paging receiver(s) 119 and peripheral device 119' in Figs. 2 and 7 as described in the specification. Further, a description of a connection of a receiver to a peripheral device as prior art is found in the United States Patents and pending applications identified on page 9 of the specification.

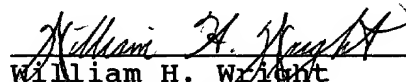
The substitute Appendix contains numbered pages which are consistent with the description of the page numbers of the Appendix on page i of the specification as amended. The copyright notices on pages 4 and 10 of the original Appendix have been deleted from pages 4 and 10 of the substitute Appendix to be consistent with the copyright notice which precedes page 1 of the original and substitute Appendix. The description of the Appendix on page i of the specification has been amended to refer to pages 10-12 as being the program for controlling the operation of the interface switch. Deleted pages 13-14 were not used as the code for controlling the interface switch.

The inventors will file a Supplemental Declaration affirming subject matter of this Amendment as being part of their invention as filed.

Please charge any shortage in the fees due in connection with the filing of this paper, including extension of time fees, to the deposit account of Antonelli, Terry, Stout & Kraus, Deposit Account No. 01-2135 (780.29767X00), and please credit any excess fees to such deposit account.

Respectfully submitted,

HENDERSON & STURM



William H. Wright
Registration No. 26,424

WHW:dlh

enter
Sub Appendix

~~APPENDIX~~

RECEIVED

Copyright Thomas Campana, Jr. 1991

define ATT_MAIL_FILE
define DELIMITER

"TFPROCK.TMP"
"End of Telefind Network Message\r"

define ATT_MAIL_FILE
define DELIMITER

```

#include <string.h>
#include <time.h>
#include <stdio.h>
#include <dos.h>
#include "safari.h"

void main(void)
{
    FILE *infile,*outfile;
    char buffer[81],chr,timestr[6],datestr[9];
    char msg_num[4];
    int msg_num_opt = 0;
    char *ptr;
    int x,day,month,line=1,attmail=0;
    time_t t;

    if ((infile = fopen(ATT_EMAIL_FILE,"rt")) == NULL)
    {
        printf("%s does not exist\n",ATT_EMAIL_FILE);
        exit(0);
    }
    if ((outfile = fopen("tmbbox.888","wt")) == NULL)
    {
        printf("Can't open TMBBOX.888\n");
        exit(0);
    }

    for(;;)
    {
        /*      get characters from .tmp file      */
        x = 0;
        do
        {
            chr = fgetc(infile);
            if (feof(infile))
            {
                fclose(infile);
                fclose(outfile);
                exit(0);
            }
            buffer[x++] = chr;
        }
        /*      until end of line      */
        while (chr != '\n' && x != 80);

        buffer[x] = '\0';      /*      terminate it      */

        if ((line == 1)
        {
            ptr = strchr(buffer,' ');
            if (ptr-buffer == 2)      /*      use 3rd character      */
            {
                sscanf(buffer,"%i[]",msg_num);
                msg_num_opt = 1;
                ptr++;
            }
            else
                ptr = buffer;

            if (*ptr == ':' && *(ptr+1) == '\0')
                attmail = 1;
        }

        if (attmail)
        {
            switch(line)

```

```

(
    case 1:
        /*      datestr = mm/dd, timestr = hh:mm      */
        sscanf(datestr,"%d/%d",&month,&day);
        /*      get year from pc      */

        t = time(NULL);
        fprintf(outfile,"Date: %s",ctime(&t));
        break;

    case 2:
        fprintf(outfile,"From: %s",buffer);
        break;

    case 3:
        fprintf(outfile,"Subject: %s",buffer);
        fprintf(outfile,"To: <name here>\n");
        if (msg_num_opt)
            fprintf(outfile,"Message %d\n",msg_num);
        break;

    default:
        fprintf(outfile,"%s",buffer);
        break;
)
else
(
    if (line == 1)
    (
        t = time(NULL);
        fprintf(outfile,"Date: %s",ctime(&t));
        fprintf(outfile,"From: tsmobex\n");
        fprintf(outfile,"Subject: Telefind Network Message\n");
        fprintf(outfile,"To: <name here>\n");
        if (msg_num_opt)
        (
            fprintf(outfile,"Message %d\n",msg_num);
            fprintf(outfile,"%s",buffer+3);
        )
        else
            fprintf(outfile,"%s",buffer);
        )
    else
        fprintf(outfile,"%s",buffer);
)

if (strcmp(buffer,DELIMITER) == 0)
(
    msg_num_opt = line = attmail = 0;
)

line++;
)

```



```

Author:      MICHAEL P. PORSCHE, SR.
              03/13/91

Program:     SAFARI3.C
Purpose:     TO EXTRACT MESSAGES FROM A TELEFIND PAGER
              VIA IN RS-232 PORT ON A PC

Compiler:    TURBO C++ 1.0
Memory Model: SMALL

```

```
*/
```

```

#include <dos.h>
#include <stdio.h>
#include <conio.h>
#include <string.h>
#include <stdlib.h>
#include "safar1.h"

```

```
/*          CONSTANTS          */
```

```

#define DTR_HI      0xd1
#define DTR_LO      0xfe
#define RTS_HI      0x02
#define RTS_LO      0xfd
#define DER_HI      0x20
#define RING_IN     0x40
#define CD_HI       0x80
#define FIVE_TICK   5
#define FIVE_SEC    96
#define TWELVE_SEC  220
#define LOG_FILE    "LOG"
#define INTRO_STRING "Please standby, retrieving messages ..."

```

```
/*          FUNCTION PROTOTYPES          */
```

```

int beep(void);
void busyoff(void);
void busyon(void);
void disoff(void);
void dison(void);
int link(void);
void print_message(void);
int rxdata(void);
int strobe(void);
int strobe_data(void);
unsigned ticks(void);
int timeout(unsigned start, int delay);

```

```
/*          VARIABLE DECLARATIONS          */
```

```

char pager_buffer[511];
int com_base, control_reg, status_reg, log_flag;
FILE *log_file;

```

```
void main(int num_arg, char **args)
```

```
{
```

```

    unsigned start;
    int restart,x;

```

```

    com_base = 0x3f8; /* use com 1 unless command line denotes otherwise */

```

```
/* get command line arguments */
```

```

/* all command line arguments begin with a single '-' and
must be separated by a single space between each other
and the program name

-1    Use COM port 1
-2    Use COM port 2
-p    Log all activity to a file named LOG */

if (num_arg > 1)
{
    for (x=1; x<num_arg; x++)
    {
        if (strcmp(args[x], "-1") == 0)
            com_base = 0x3f8;
        if (strcmp(args[x], "-2") == 0)
            com_base = 0x2f8;
        if (strcmp(args[x], "-p") == 0)
            log_flag = 1;
    }
}

if (log_flag)
    if ((log_file = fopen(LOG_FILE, "at")) == NULL)
        printf("Unable to open LOG\n");

control_reg = com_base + 4;
status_reg = com_base + 6;

clrscr();

if (link() == 0) /* is pager attached ? */
{
    printf("Please attach Message Receiver \n");
    exit(0);
}

busyon(); /* start busy at logic high */

if (log_flag)
    fprintf(log_file, "Initiating process \n");
printf("%s\n", INTRO_STRING);
dison(); /* push display button */
sleep(2);
do
{
    start = ticks();
    restart = 0;
    do
    {
        if (beep())
        {
            print_message();
            restart = 1;
            start += TWELVE_SEC;
            break;
        }
    }
    /* hold display button for 12 seconds */
    while(1 timeout(start, TWELVE_SEC));
}
while(restart);

disoff(); /* release the display button */
if (log_flag)
{
    fprintf(log_file, "Process Complete \n");
}

```

```

        fclose(log_file);
    }

}

/*      pager beep      */
int beep(void)
{
    /*      accesses the RI line via the Status Register
        which is activated when the pager beeps      */

    unsigned start;

    start = ticks();
    while ( ! timeout(start,FIVE_TICK))
    {
        if (((inputb(status_reg) & RING_IN) == 0) )
            return(1);
    }
    return(0);
}

/*      busyon & busyoff toggle the DTR line via the
    Control Register to strobe in data from the pager      */

void busyoff(void)
{
    outputb(control_reg, (inputb(control_reg) | DTR_HI));
}

void busyon(void)
{
    outputb(control_reg, (inputb(control_reg) & DTR_LO));
}

/*      diaon & diaoff toggle the RTS line via the Control Register
    to simulate the pressing of the display button on the pager      */

void diaon(void)
{
    outputb(control_reg, (inputb(control_reg) | RTS_HI));
}

void diaoff(void)
{
    outputb(control_reg, (inputb(control_reg) & RTS_LO));
}

int link(void)
{
    /*      accesses the CD line via the Status Register
        which is logic high when pager is connected      */

    if (((inputb(status_reg) & CD_HI) == 0) )
        return(0);
    return(1);
}

void print_message(void)
{
    FILE *file;
    unsigned start;
    int x,y=0,x=0,chr,bit;

```

```

busyoff(); /* ready to accept pager data */
/* read until end code received */
while (chr != 3)
{
    shr = 0;
    start = tictac();

    /* wait for start bit */

    do
    {
        bit = strobe();
        if (bit == 0)
            break;
    }
    while (!timeout(start, FIVE_SEC));

    if (bit)
    {
        if (log_flag)
            fprintf(log_file, "Transmission Error, recheck connection\n");
        disoff();
        exit(0);
    }

    /* strobe out 8 bit data */

    for (x=1; x<9; x++)
    {
        chr <<= 1;
        chr |= bit = strobe_data();
    }

    /* clear out stop bits */
    for (x=1; x<3; x++)
    {
        strobe_data();
    }

    /* extract start and end codes from message

    pager signon      02, 10, 00, 33
    pager signoff     03 */

    if ((y > 3) && (chr != 3))
    {
        /* pager characters 96 and 97 are converted to
        0xfa and 0xfb to display on pager */

        if (chr == 0xfa) /* convert to CR */
            chr = '\n';
        if (chr == 0xfb) /* convert to TAB */
            chr = 0x09;

        pager_buffer[z] = chr;
        z++;
    }
    y++;
}

pager_buffer[z] = '\0'; /* null terminate */
busyon(); /* finished receiving data */

```

```

    if (log_flag)
        fprintf(log_file, "%s\n", pager_buffer);

    if ((file = fopen(ATT_EMAIL_FILE, "at")) == NULL)
        fprintf(log_file, "Unable to open TFMBOX.TMP\n");
    else
    {
        fprintf(file, "%s\n", pager_buffer);
        fprintf(file, "%s", DELIMITER);
        fclose(file);
    }

    start = ticks();
    while(timeout(start, FIVE_SEC))
    {
        /* wait for erase beep */
        if (beep()) break;
    }
    sleep(1); /* wait one more second */
}

int rxdata(void)
{
    /* accesses the DSR line via the Status Register
       which returns the byte value */

    if (inportb(status_reg) & DSR_M1)
        return(0);
    return(1);
}

int strobe(void)
{
    int bit;

    busyon();
    delay(1);
    busyoff();
    delay(4);
    bit = rxdata();
    return(bit);
}

int strobe_data(void)
{
    int bit;

    busyon();
    delay(2);
    bit = rxdata();
    busyoff();
    delay(1);
    return(bit);
}

unsigned ticks(void)
{
    /* returns timer ticks (approx. 18.2/sec)
       using only lower registers */

    union REGS in, out;

    in.x.ax = 0x0;
    int86(0x1a, &in, &out);
    return(out.x.dx);
}

```

```

    }
    int timeout(unsigned start, int delay)
    {
        /*      used for timing events of up to approx. 1 hour.
           used in conjunction w/ticks()      */

        unsigned current;

        current = ticks();
        if (start <= current && (start + delay) < current)
            return(1);
        if (start > current && (start - 65535 + delay) < current)
            return(1);
        return(0);
    }

```

```

1: 00000000
2: 00000000
3: 00000000
4: 00000000
5: 00000000
6: 00000000
7: 00000000
8: 00000000
9: 00000000
10: 00000000
11: 00000000
12: 00000000
13: 00000000
14: 00000000
15: 00000000
16: 00000000
17: 00000000
18: 00000000
19: 00000000
20: 00000000
21: 00000000
22: 00000000
23: 00000000
24: 00000000
25: 00000000
26: 00000000
27: 00000000
28: 00000000
29: 00000000
30: 00000000
31: 00000000
32: 00000000
33: 00000000
34: 00000000
35: 00000000
36: 00000000
37: 00000000
38: 00000000
39: 00000000
40: 00000000
41: 00000000
42: 00000000
43: 00000000
44: 00000000
45: 00000000
46: 00000000
47: 00000000
48: 00000000
49: 00000000
50: 00000000
51: 00000000
52: 00000000
53: 00000000
54: 00000000
55: 00000000
56: 00000000
57: 00000000
58: 00000000
59: 00000000
60: 00000000
61: 00000000
62: 00000000
63: 00000000
64: 00000000
65: 00000000
66: 00000000
67: 00000000
68: 00000000
69: 00000000
70: 00000000
71: 00000000
72: 00000000
73: 00000000
74: 00000000
75: 00000000
76: 00000000
77: 00000000
78: 00000000
79: 00000000
80: 00000000
81: 00000000
82: 00000000
83: 00000000
84: 00000000
85: 00000000
86: 00000000
87: 00000000
88: 00000000
89: 00000000
90: 00000000
91: 00000000
92: 00000000
93: 00000000
94: 00000000
95: 00000000
96: 00000000
97: 00000000
98: 00000000
99: 00000000
100: 00000000

```

```

        /* mark the end of the command line you built,so you can add ending
           delimiter */
        sys_command[i] = NULL;
        /* add the ending quote for the users message so shell wont
           interepert special characters */
        strcat(sys_command, "'");
        /* execute command you built */
        system(sys_command);

        printf("sending message: %s\n", sys_command);
    }
    else {
        if(strlen(msg) == 0 ) {
            return(0);
        }
        /* print error for invalid message length */
        printf("telemail error: invalid message length: %s\n", msg);
        return(0);
    }

    return(i);
}

/*****
 *
 * function: getline(hold-buffer, input-file-pointer)
 * arguments: pointer to buffer where line read will be heald,
 *             file pointer to input file
 * description: reads 1 line of text from the input line and stores the
 *              line read into the buffer passed.
 * returns: -1 if EOF or number of characters read in
 *****/
getline(buff, fp)
char *buff;
FILE *fp;
{
    int ch, cnt;

    /* keep on reading characetr from file so long as end of file not
       reached or char is the end of line */
    for(cnt = 0; ((ch = fgetc(fp)) != EOF) && ch != '\n'; cnt++) {
        /* MOD BY OT 11/29/90 convert tab to space */
        /* convert tabs to single space */
        if(ch == 9) {
            ch = ' ';
        }
        /* MOD BY OT 11/29/90 dont allow control char */
        /* only load in ascii characters */
        if(isprint(ch) != 0) {
            buff[cnt] = ch;
        }
        else {
            /* turn control characters to spaces */
            buff[cnt] = ' ';
        }
    }

    /* mark the end of the buffer you built */
    buff[cnt] = '\0';
}

```

```

/* *****
*
* function: send_mesg(message-pointer)
* arguments: pointer to text message(capcode,text) to be sent
* description: takes passed message text makes sure the first 8 positions
*               are numeric(capcode). it builds and executes the network
*               send command(netsend.sh) to sedn the message passed.
* returns: 0 if not sent otherwise the number of characters sent out
*
***** */
int send_mesg(mesg)
char *mesg;
{
    char sys_command[700];
    int i;
    int ch;
    char *mesg_ptr;

    /* left justify the message passed to remove leading spaces */
    strljust(mesg, 512);
    /* trim off trailing blank spaces from the message */
    strtrim(mesg);

    /* make sure you have a capcode at least */
    if(strlen(mesg) > 8) {

        /* start to build the command to be executed to send message retrieved
           from the mail box */
        strcpy(sys_command, "netsend.sh ");

        /* loop while still more characters in the message */
        for(mesg_ptr = mesg, i = 11; *mesg_ptr != NULL; i++, mesg_ptr++) {

            /* make sure the first 8 positions of the message are numeric */
            if((i < 19) && (*mesg_ptr < '0' || *mesg_ptr > '9')) {
                printf("telemail error: invalid capcode: %s\n", mesg);
                return 0;
            }

            /* is the user didnt separete capcode & message then insert a
               space into the command */
            if(i == 19 && *mesg_ptr != ' ') {
                sys_command[19] = ' ';
                i = 20;
            }

            /* enclose the users message with ' so shell wont interpret
               special characters */
            if(i == 20) {
                sys_command[20] = '\\';
                i = 21;
            }

            /* put the character from the message onto to the
               command to be executed */
            sys_command[i] = *mesg_ptr;
        }
    }
}

```



```

/* since your just starting clear the message area */
memset(mesg, NULL, MAXMSGLEN);

/* keep on getting lines from the file until you reach end of file */
while(getline(buff, fp) != -1) {
    /* every mail message start with the word "From " */
    if(strncmp(buff, "From ", 5) == 0) {
        /* set flag telling you are currently going thru mail header
           so you dont add it to the message */
        in_header = 1;
        /* call routine to the last message if any exists */
        send_mesg(mesg);
        continue;
    }

    /* a mail header end with the following string */
    if(strncmp(buff, "Content-Length:", 15) == 0) {
        /* turn off flag so you know you are no longer in mail
           message header */
        in_header = 0;
        /* clear the old message since this is a new one */
        memset(mesg, NULL, MAXMSGLEN);
        continue;
    }

    /* if the line you are now reading in not part of the mail header
       add it to the message */
    if(in_header == 0) {
        strljust(buff, 512);
        strtrim(buff);
        /* make sure you dont add more than the message length */
        if( (strlen(buff) + strlen(mesg)) < MAXMSGLEN) {
            strcat(mesg, " ");
            strcat(mesg, buff);
        }
    }
} /* end of read line while */

/* send the last message in the file */
send_mesg(mesg);
}

```